Bài tập thực hành 3

I. Lý thuyết.

2. Độ phức tạp của thuật toán đệ quy (Công thức và ý nghĩa của từng tham số)

$$f(x) = \begin{cases} c,\ if\ n\ =\ n_0 \\ a.T(f(n))\ +\ g(n)\ otherwise \end{cases}$$

Giải thích tham số:

c: Thời gian chạy cho trường hợp cơ sở

n_0: Cơ sở đệ quy.

a: Số lần gọi đệ quy.

n trong f(n) : kích cỡ vấn đề được xử lý bởi gọi đệ quy.

g(n): Tất cả những quá trình khác

3. **Phương pháp giải công thức truy hồi (Nêu rõ phương pháp, ví dụ và cách giải ví dụ dựa trên các phương pháp đã nêu)**

1. Phương pháp thay thế lặp lại

Tùy vào từng bài toán sẽ có cách giải khác nhau nhưng tổng quát sẽ có các bước sau:

- Xác định quan hệ truy hồi của thuật toán.
- Giải quan hệ truy hồi trên 1 số lượng nhỏ các giá trị của biến đầu vào để tìm ra một mẫu.
- Giả sử mẫu tìm được có đúng với tất cả các giá trị của biến đầu vào.
- Chứng minh mẫu tìm được bằng cách sử dụng nguyên lý toán học.
- Sử dụng mẫu để tính toán độ phức tạp trên tất cả các giá trị của biến đầu vào.

Ví dụ:

$$f(x) = \begin{cases} 1\ if\ n\ =\ 1 \\ T_S(n-1) +\ 1\ if\ n\ >\ 1 \end{cases}$$

Giải:

$T_S(n) \quad = T_S(n - 1) + 1$

$= T_S(n - 2) + 1 + 1$

$= T_S(n - 3) + 1 + 1 + 1$

$= ...$

$= T_S(n – (n - 1)) + 1 + ... + 1$ (n - 1 lần)

$= 1 + 1 + ... + 1 = n$

$=> T_S(n) = O(n)$

2. Phương pháp lý thuyết chung

Kết quả của:

$$f(x) = \begin{cases} 1 \; if \; n \le 1 \\ aT(n / b) + n^d \; if \; n > 1, \; a \ge 1, b > 1, d \ge 0 \end{cases}$$

Là:

$$T(n) = \begin{cases} O(n^d) \; if \; a < b^d \\ O(n^d logn) \; if \; a = b^d \\ O((n^{\log_b a}) \; if \; a > b^d \end{cases}$$

Ví dụ: T(n) = 2T(n / 2) + n

a = 2, b = 2, d = 1 => $a = b^d$ => T(n) = O(nlogn)

II. Giải công thức đệ quy để xác định độ phức tạp thuật toán.

1. $$T(n) = \begin{cases} 1 \; if \; n = 0, n = 1 \\ T(n - 2) + 2 \; n > 1 \end{cases}$$

Giải:

T(n) = T(n – 2) + 2

= T (n – 4) + 2 + 2

= T(n – 6) + 2 + 2 + 2

= ...

= T(n – 2 *(n / 2)) + (2 + 2 + ... 2) (n/2 lần)

$= T(n - 2 *(n / 2)) + 2 * (n /2)$

$= n$

$\Rightarrow T(n) = O(n)$